SeqL+: Secure Scan-Obfuscation with Theoretical and Empirical Validation

Seetal Potluri, *Member, IEEE*, Shamik Kundu, *Student Member, IEEE*, Akash Kumar, *Senior Member, IEEE*, Kanad Basu, *Senior Member, IEEE*, and Aydin Aysu, *Senior Member, IEEE*.

***Abstract*—Scan-obfuscation is a powerful methodology to protect Silicon-based intellectual property from theft. Prior work on scan-obfuscation in the context of logic-locking have unique limitations, which are addressed by our previous work, *SeqL*, which looks at functional output corruption to obfuscate scan-chains, but is unable to resist removal attacks on circuits with inadequate number of flip-flops without feedback. To address this issue, we propose to scramble flip-flops with feedback to increase key-length without introducing further vulnerabilities. This study reveals the first formulation and complexity analysis of Boolean Satisfiability (SAT)-based attack on scan-scrambling. We formulate the attack as a conjunctive normal form (CNF) using a worst-case $\mathcal{O}(n^3)$ reduction in terms of scramble-graph size n. In order to defeat SAT-based attack, we propose an iterative swapping-based scan-cell scrambling algorithm that has $\mathcal{O}(n)$ implementation time-complexity and $\mathcal{O}(2^{\lfloor \frac{\alpha.n+1}{3} \rfloor})$ SAT-decryption time-complexity in terms of a user-configurable cost constraint α $(0 < \alpha \leq 1)$.**

***Index Terms*—IP Piracy, Scan-chains, Scan-scrambling.**

I. Introduction

Scan-obfuscation is a powerful methodology and it has been recently applied to defend logic-locking attacks on sequential circuits [1]–[10]. These techniques have unique limitations, including inability to handle reverse engineering [1], vulnerability to ScanSAT [3], and increasing layout complexity [4]. Our previous work, *SeqL* [11], addresses all these issues but is unable to resist removal attacks on circuits without adequate number of flip-flops without feedback (R_{wof}). To address this concern, we propose *SeqL+*, a secure and scalable scan-scrambling approach for flip-flops with feedback.

In order to launch ScanSAT [3], the adversary needs to know the ordering of scan flip-flops (SFFs) in the scan-chain in order to initialize them to known-values, and observe the corresponding next-state responses. Since *scan-scrambling* hides the ordering of SFFs in the scan-chain, the attacker is unable to achieve this, thus preventing direct applicability of SAT-based attack.

In ScanSAT [3], the authors consider the various inputs to the scramble-MUX coming from different scan-chains. Since the attacker knows that the correct input to the scramble-MUX comes from the same scan-chain (which is unique), it

S. Potluri and A. Aysu are with the Electrical and Computer Engineering Department, North Carolina State University, Raleigh, NC, 27606.
S. Kundu and K. Basu are with the Department of Electrical and Computer Engineering, University of Texas at Dallas, Richardson, TX, 75080.
A. Kumar is with the Department of Computer Science, Technical University of Dresden, 01062 Dresden, Germany

Manuscript received September, 2021; revised December, 2021; revised February, 2022; revised April, 2022; revised July, 2022.

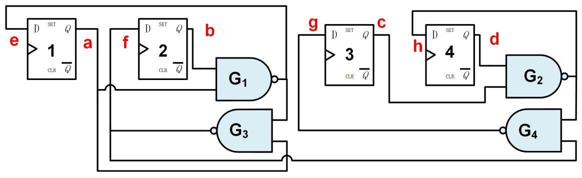

Fig. 1. Sample circuit consisting of four gates and four flip-flops.

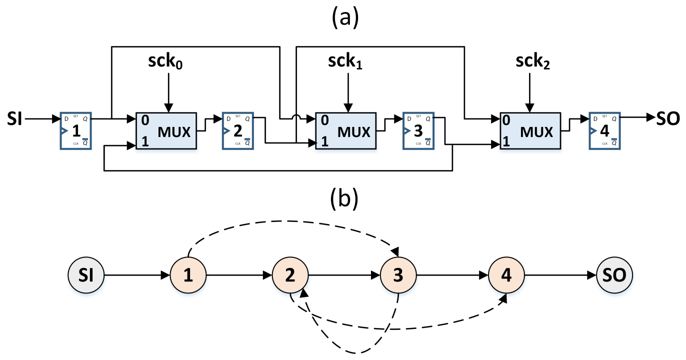

Fig. 2. (a) Sample scrambled scan-chain corresponding to Figure 1 and (b) Corresponding *scramble-digraph*

is impractical/insecure. Hence, in *SeqL+* we consider all the inputs to the scramble-MUX come from the same scan-chain and proceed with the security analysis. Moreover, the existing scan obfuscation approaches have been recently shown to be vulnerable to new attacks [3], [12], [13]. The novelty of our work is to use scrambling of scan flip-flops, so as to exponentially increase the number of equivalence classes to defend SAT attack, and without introducing further vulnerabilities.

II. SeqL+: Scrambling for designs with small R_{wof}

The key research question is: **"Would scan-scrambling form equivalence classes (ECs) like conventional, combinational logic-locking, causing a vulnerability against SAT-based attacks?"**. This subsection conducts complexity analysis and formulation of scan-scrambling against such attacks, and proves crucial properties of ECs.

Graph-based Formulation: Every scan-scrambled instance can be formulated as a digraph $G = (V, E)$ where (i) Scan-input (SI), all $SFFs$ and scan-output (SO) are represented as vertices (V) in G; and; (ii) The connections between SI, $SFFs$ and SO in the circuit are represented as directed edges (E) between corresponding vertices in G, where the direction signifies the signal flow. A *Hamiltonian path* in a digraph is a path that visits each vertex exactly once.

Figure 1 shows a sample circuit with 4 2-input nand gates and 4 flip-flops (prior to scan-insertion). Figure 2(a) shows an example of scrambled scan-chain corresponding to this circuit, and Figure 2(b) shows the corresponding *scramble-digraph*. There are 2 possible *Hamiltonian Paths* (HPs) in Figure 2(b), $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ corresponding to $\{sck_0, sck_1, sck_2\} = (011)_2$ and $1 \rightarrow 3 \rightarrow 2 \rightarrow 4$ corresponding to $\{sck_0, sck_1, sck_2\} = (100)_2$. The scramble-keys corresponding to the *HPs* in G ensure that all the $SFFs$ are connected together along with SI and SO to form the scan-chain. The remaining scramble-key-combinations disassociate some of the $SFFs$ from the scan-chain.

Theorem II.1. *Every* HP *in G has* injective mapping *to exactly one* valid *scramble-key-combination.*

Proof. Let H_s be a selected *HP* in G. Now, H_s corresponds to a particular ordering of vertices in G, say $\{v_1(H_s), v_2(H_s) \dots v_N(H_s)\}$. Since each vertex in G has *injective mapping* to a unique SFF in the circuit, H_s corresponds to a unique ordering of SFFs, say $\{SFF_1(H_s), SFF_2(H_s) \dots SFF_N(H_s)\}$.

Let $SFF_i(H_s)$ be the i^{th} scan flip-flop and let $k_i(H_s)$ be scramble-key-bit corresponding to the scramble-MUX at the input of scan flip-flop $SFF_i(H_s)$:

- **Basis step**: $SFF_1(H_s)$ is the first scan flip-flop in the scan-chain, which means SI drives $SFF_1(H_s)$. To achieve this, there must be a unique assignment to $k_1(H_s)$. Hence, key-bit uniqueness is true for *i=1*.
- **Induction step**: Assume scramble-key-bit uniqueness is true for *i=l*. $SFF_{l+1}(H_s)$ is the $(l+1)^{th}$ scan flip-flop, which means output of $SFF_l(H_s)$ should drive input of $SFF_{l+1}(H_s)$. In order to achieve this, there is a unique assignment to $k_{l+1}(H_s)$. Thus, key-bit uniqueness is true for *i=l+1*.

Hence, by *finite induction* we infer all scrambling-key-bits are unique for *HP* H_s. This indicates each *HP* in G corresponds to exactly one *scramble-key*, and since the converse is also true, the mapping is *injective*, thus the proof. QED

A. Attacking scan-scrambling using SAT formulation

The formulation comprises multiple constraints:

1) Hamiltonian Path (HP) Constraints: So far, we have seen how to break scrambling using HP search, next we shall see how to break using SAT-based attack.

Formulation: Given a scramble digraph G, we construct a Boolean CNF $B(G)$ such that such that $B(G)$ is satisfiable iff G has a HP. $B(G)$ has n^2 Boolean variables $\{x_{ij}\}$, $1 \leq i, j \leq n$. A satisfying truth assignment to $B(G)$ does provide us with a HP for G. Here, x_{ij} means the i^{th} position in the HP is occupied by node-j. An HP can be expressed as a permutation π of $\{1, 2, \dots n\}$, where:

- $\pi(i) = j \Rightarrow i^{th}$ position is occupied by node-j.
- $(\pi(i), \pi(i+1)) \in G$ for $i = 1, 2, \dots (n-1)$

Considering the example motivated thus far, $n = 4$, hence $B(G)$ has $4^2 = 16$ variables $\{x_{ij}\}$, $1 \leq i, j \leq 4$. The *Hamiltonicity Clausebase* is produced using HP constraints, which are multiple-fold:

1) Each node j must appear in the path, $1 \leq j \leq n = 4$
   - $x_{1j} \vee x_{2j} \vee x_{3j} \vee x_{4j}$

   Thus, total # constraints in this category is n.
2) No node j appears twice in the path, $1 \leq j \leq n = 4$
   - $\neg x_{1j} \vee \neg x_{2j}$, $\neg x_{1j} \vee \neg x_{3j}$, $\neg x_{1j} \vee \neg x_{4j}$
   - $\neg x_{2j} \vee \neg x_{3j}$, $\neg x_{2j} \vee \neg x_{4j}$, $\neg x_{3j} \vee \neg x_{4j}$

   Thus, total # constraints in this category is $\binom{n}{2} \times n$.
3) Every position i on the path must be occupied, $1 \leq i \leq n = 4$
   - $x_{i1} \vee x_{i2} \vee x_{i3} \vee x_{i4}$

   Thus, total # constraints in this category is n.
4) No two nodes j and k occupy the same position i in the path, $1 \leq i, j, k \leq n = 4, j \neq k$
   - $\neg x_{i1} \vee \neg x_{i2}$, $\neg x_{i1} \vee \neg x_{i3}$, $\neg x_{i1} \vee \neg x_{i4}$
   - $\neg x_{i2} \vee \neg x_{i3}$, $\neg x_{i2} \vee \neg x_{i4}$, $\neg x_{i3} \vee \neg x_{i4}$

   Thus, total # constraints in this category is $\binom{n}{2} \times n$.
5) Non-adjacent nodes i and j cannot be adjacent in the path, $1 \leq i, j \leq n = 4$
   - $\neg x_{1i} \vee \neg x_{2j}$, $\neg x_{2i} \vee \neg x_{3j}$, $\neg x_{3i} \vee \neg x_{4j}$

Let's denote the set of clauses in this category as C_{HP}.

2) Constraints connecting SI/SO bits to the SFF outputs/inputs respectively: Considering the example motivated thus far, since $n = 4$, let I_1, I_2, I_3, I_4 be the input bits applied serially through SI and a, b, c, d are the outputs of SFFs $1, 2, 3, 4$ respectively as shown in Figure 1. The constraint connecting SI bits to SFF output a can be formulated as follows:

- $a = x_{11}.I_1 + x_{12}.I_2 + x_{13}.I_3 + x_{14}.I_4$

The remaining SFF outputs b, c, d can also be likewise expressed as a constraint connecting SI bits. Similar relationship exists between the e, f, g, h (the inputs of flip-flops as shown in Figure 1) and O_1, O_2, O_3, O_4 i.e. the output bits serially scanned out through SO.

Let's denote the set of clauses in these categories as C_I and C_O respectively. Each constraint corresponds to one SFF and there are n SFFs. Further, each constraint is a function of n 2-input and gates and $(n-1)$ 2-input or gates. Since a 2-input and gate as well as a 2-input or gate translates to 3 clauses each in the CNF, there are altogether $3 \times (n + (n-1)) = 3 \times (2n-1)$ clauses, or in other words, $|C_I| = |C_O| = 3 \times (2n-1)$.

3) Combinational Circuit Constraints: Fig. 1 shows 4 2-input nand gates G_1, G_2, G_3, G_4 in the combinational portion of the scan-scrambled circuit. After converting them to clauses, let the obtained set of clauses in this category as C_{Combo}.

4) Running SAT-based attack on Scan-Scrambling: Using the reverse-engineered netlist, the adversary computes the clausebases corresponding to HP constraints C_{HP}, connection constraints C_I, C_O, and combinational circuit constraints C_{Combo}. The adversary subsequently merges these clausebases to produce the original scramble CNF $B(G)$ needed to attack scan-scrambling:

$$B(G) = C_{HP} \cup C_I \cup C_O \cup C_{Combo} \tag{1}$$

The adversary uses this scramble CNF $B(G)$, and runs the SAT-based attack to solve for $\vec{X}$. Considering the example

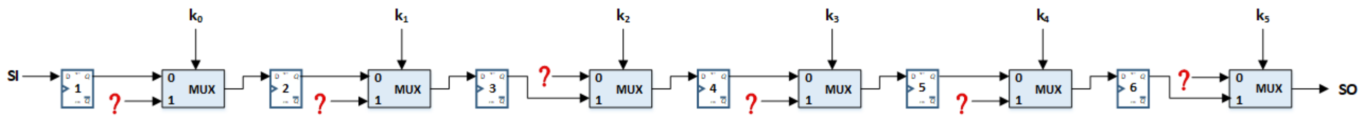

Fig. 3. One of the inputs of each scramble-MUX is known. The second input is unknown and has to be decided in such a way, so as to maximize the number of *HPs* in the resultant scramble-graph. The correct EC is $\{(k_0, , k_1, k_2, k_3, k_4, k_5)\} = 001001$.

motivated thus far, the SAT-based attack returns $x_{11} = x_{23} = x_{32} = x_{44} = 1$ and $x_{ij} = 0$ otherwise. This corresponds to $\pi(1) = 1$, $\pi(2) = 3$, $\pi(3) = 2$, $\pi(4) = 4$, or in other words the HP $(1 \rightarrow 3 \rightarrow 2 \rightarrow 4)$. The adversary then uses $\vec{X}$ to *decrypt* the scrambling-key by looking at the reverse-engineered netlist. Although we have motivated using a sample circuit, the concept is generic and hence can be extended to any arbitrary scan-scrambled circuit. The next section discusses our proposed defense against SAT-based attack on scrambling.

B. Defending SAT-based attack on Scan-Scrambling

It is well-known that SAT-based attack is a brute-force search on the ECs [14]. The goal of our defense is to increase the number of scramble ECs, to make the attack computationally infeasible. Based on Theorem II.1, this translates to increasing the number of *HPs* in G. Thus, the **objective** is to connect the second input to the scramble-MUXes to produce a *scramble-digraph* G with maximum number of *HPs*.

1) Search Space Exploration: We assume only *security scan-chain* is scrambled, whose length is n. We assume a scramble-MUX at the input of each SFF as well as the scan-out port, thus there is a total of $(n+1)$ scramble-MUXes, as shown in Figure 3. Since one of the inputs to each scramble-MUX is fixed corresponding to the correct scramble, and the second input available for exploration, the designer needs to evaluate the search space and decide the best choice. Avoiding self-loops and repetition, the second input of each scramble-MUX can be connected in $(n-1)$ ways. Thus, size of the scrambling search space is $(n-1)^{(n+1)}$.

If we define $\delta_i = |c_i^1 - c_i^2|$ as the disturbance produced on vertex-i, where c_i^1 and c_i^2 be the indices of the vertices whose outputs are connected to the first and second inputs the corresponding scramble-MUX, and the disturbance vector $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$, then:

$$|\Delta| = \sqrt{\sum_i \delta_i^2} \tag{2}$$

We have performed a *brute-force search* and checked the distribution of $|\Delta|$ for $\eta = 3, 4, 5, 6$ when running a *brute-force search*. The lowest value of $|\Delta|$ was observed as 6, 10, 11 and 12 for $\eta = 3, 4, 5, 6$ respectively. We have verified this corresponds to the adjacent-scrambling (AS) case across all values of η. Similar pattern was observed for higher values of η, thus demonstrating the power of adjacent-scrambling. Since it is not possible to perform *brute-force search* for higher values of η, we exploit this observation to defend SAT-based attack using adjacent-scrambling. Algorithm 1 shows the proposed AS algorithm, where C is the circuit and η is the user-defined cost/area constraint ($0 < \alpha = \frac{\eta}{n} \leq 1$). The SFFs are allowed to be permuted only once, and it is also not allowed to permute their fanout SFFs as well, once permuted.

```
Algorithm 1: Iterative Swapping-based Scrambling
  Input: $C$, $\eta$
1 Create a scramble-digraph $G = (V, E)$ with SFFs as
    vertices, and directed edges corresponding to signal
    flow in $C$;
2 $C' = C, n_s \rightarrow 0$;
3 $G' = G$;
4 while $n_s \leq \eta$ do
5     Mark $\{v_{n_s}, v_{n_s+1}, v_{n_s+2}\}$ as visited ;
6     Scramble SFFs $\{v_{n_s}, v_{n_s+1}\}$ and add directed
        edges to $G'$ corresponding to the additional signal
        flow in $C'$;
7     $n_s \rightarrow n_s + 3$;
  Result: $C'$
```

Since it is a single loop iterating over the SFFs until the cost constraint η is met, the algorithm time-complexity is $\mathcal{O}(\eta)$.

III. Experimental Evaluation

In this section, we (a) compare the security and overheads of *SeqL+* with prior obfuscation schemes and (b) demonstrate the resilience of *adjacent scrambling* against SAT-based attack. Table I compares SeqL+ with EFF [2], dynamically obfuscated scan (DOS) [8], SeqL [11], robust design-for-security (RDFS) [9] and key-trapped design-for-security (kt-DFS) [10] in terms of resilience to various attacks and the area overheads. For large circuits, since $< 1\%$ of the flip-flops are scrambled, they will be chosen on the non-critical timing paths, so no timing overhead. Similarly, since the multiplexer appears only during scan mode of operation, there is no power overhead. It can be seen that SeqL+ is most secure, and the overheads are better than *EFF*, *DOS*, *SeqL*, *RDFS*, and *kt-DFS* with increase in circuit size.

A. Resilience of adjacent scrambling to SAT-Attack

Theorem III.1. *The number of* scramble ECs *produced using the* adjacent scrambling *algorithm is* $2^{\lfloor \frac{\eta+1}{3} \rfloor}$.

Proof. Algorithm 1 swaps/scrambles two adjacent vertices per iteration in the graph G consisting of $(n+1)$ vertices in G. For every scramble-pair (u, v), 3 vertices get eliminated in each iteration. and creates 2 *valid paths* ($u \rightarrow v$ and $v \rightarrow u$ in Algorithm 1). Thus, Algorithm 1 iterates $\lfloor \frac{\eta+1}{3} \rfloor$ times, defined by a cost constraint $\eta = \alpha.n$. Since each iteration decides 3 successive positions in the permutation, and the positions-under-scrutiny are mutually exclusive across iterations, the number of *HPs* multiply *geometrically*. Thus, the number of *HPs* produced after iteration-(i) is 2^i. Since prior to termination, Algorithm 1 iterates $\lfloor \frac{\eta+1}{3} \rfloor$ times, *HPs*

TABLE I
COMPARISON OF SECURITY AND OVERHEADS. WE ASSUME IN SEQL, ALL FLIP-FLOPS WITHOUT FEEDBACK ARE LOCKED. IN SEQL+, WE SCRAMBLE 100 FLIP-FLOPS FOR REMOVAL SECURITY.

Bench.	*# ScanSAT-res.* [3]			*# Removal-res.*		*# SaLa-res.* [12]		*# NNgSAT-res.* [13]		*# Overhead*						
	EFF [2]	DOS [8]	SeqL+	SeqL [11]	SeqL+	RDFS [9]	SeqL+	kt-DFS [10]	SeqL+	EFF [2]	DOS [8]	SeqL [11]	RDFS [9]	kt-DFS [10]	α	SeqL+
b04	✖	✖	✓	✖	✓	✖	✓	✖	✓	8%	27.6%	2.6%	1.3×	40.4%	1	27.3%
b12	✖	✖	✓	✖	✓	✖	✓	✖	✓	10%	28.2%	0.7%	1.4×	41.3%	0.8	14%
b18	✖	✖	✓	✖	✓	✖	✓	✖	✓	3.8%	0.26%	0.07%	1.2%	0.43%	0.03	0.03%
b19	✖	✖	✓	✖	✓	✖	✓	✖	✓	3.7%	0.13%	0.03%	0.63%	0.19%	0.015	0.01%
b20	✖	✖	✓	✖	✓	✖	✓	✖	✓	3.3%	1.5%	0.3%	7.2%	2.2%	0.2	0.3%
b21	✖	✖	✓	✖	✓	✖	✓	✖	✓	3.2%	1.5%	0.3%	7.1%	2.14%	0.2	0.3%
b22	✖	✖	✓	✖	✓	✖	✓	✖	✓	3.3%	1%	0.2%	4.9%	1.5%	0.14	0.14%

TABLE II
CNF AND SAT-BASED ATTACK STATISTICS FOR *SeqL+* OBFUSCATED CIRCUITS WITH $\#SFFs > 50$ BUT $R_{wof} < 50$. ALGORITHM 1 WAS USED FOR SCRAMBLING THE SCAN-CHAINS.

Bench.	$SFFs$	R_{wof}	Ite. Dec. (τ_0)	*# Lit.*	*# Clauses*			*#Iters.* (# eq. cls.)	Est. Tot. Dec. time (τ)
					$\lvert C_{HP}\rvert$	$\lvert C_I \cup C_O\rvert$	$\lvert C_{Combo}\rvert$	$2^{\lfloor \frac{n+1}{3} \rfloor}$	$\tau_0 * 2^{\lfloor \frac{n+1}{3} \rfloor}$
b04	66	8	51 s	$10^{4.0}$	$10^{5.7}$	$10^{2.9}$	$10^{3.3}$	$10^{6.62}$	6.7 **years**
b12	121	6	173 s	$10^{4.5}$	$10^{6.5}$	$10^{3.2}$	$10^{3.5}$	$10^{12.04}$	$10^{6.8}$ **years**
b13	53	10	43 s	$10^{3.8}$	$10^{5.5}$	$10^{2.8}$	$10^{3.0}$	$10^{5.42}$	131 **days**
b18	3,320	23	53 $hrs.$	$10^{6.3}$	$10^{9.3}$	$10^{4.1}$	$10^{5.5}$	$10^{100.2}$	$10^{97.9}$ **years**
b19	6,642	30	91 $hrs.$	$10^{6.4}$	$10^{9.3}$	$10^{4.1}$	$10^{5.8}$	$10^{100.2}$	$10^{98.2}$ **years**
b20	490	22	7 $min.$	$10^{5.7}$	$10^{8.4}$	$10^{3.8}$	$10^{4.8}$	$10^{49.1}$	$10^{44.2}$ **years**
b21	490	22	6 $min.$	$10^{5.7}$	$10^{8.4}$	$10^{3.8}$	$10^{4.8}$	$10^{49.1}$	$10^{44.1}$ **years**
b22	735	22	11 $min.$	$10^{6.1}$	$10^{8.9}$	$10^{4.0}$	$10^{5.0}$	$10^{73.8}$	$10^{69.1}$ **years**

in the scramble graph produced through *adjacent scrambling* is $2^{\#iterations} = 2^{\lfloor \frac{\eta+1}{3} \rfloor}$, thus the proof. QED

B. Complexity Analysis

Based on *Theorem III.1*, the number of iterations the while loop in SAT-based attack [14] executes is equal to the number of *scramble ECs*= $2^{\lfloor \frac{\eta+1}{3} \rfloor}$, thus ensuring $\mathcal{O}(2^{\lfloor \frac{\eta+1}{3} \rfloor})$ SAT-decryption time-complexity. In industry practice, for large processors, typically maximum scan-chain-length (n) is typically around $500 - 1000$, thus the SAT-attack complexity can be made arbitrarily large as shown in Table II, making it practically impossible to decrypt the scrambling-key. Hence adjacent scrambling is computationally-secure against SAT-based attack.

Since by definition, *adjacent scrambling algorithm* swaps adjacent nodes, there are altogether $(n-1)+(\eta-1) = n+\eta-2$ adjacent node-pairs in the scramble-graph. All the remaining node-pairs in the complete digraph are non-adjacent, which equals $2 \times \binom{n}{2} - (n+\eta-2) = n^2 - 2n - \eta + 2$. For each non-adjacent node-pair, there are $(n-1)$ possible ways to be placed adjacent to the path, so altogether the number of non-adjacent node constraints are:

$$(n^2-2n-\eta+2)\times(n-1) = n^3-2n^2-\eta.n+2n-n^2+2n+\eta-2$$
$$= n^3 - 3n^2 + 4n - \eta.n + \eta - 2 \quad (3)$$

Substituting this in the results from section II-A1, we get

$$|C_{HP}| = 2n \times \left(1+\binom{n}{2}\right) + (n^3 - 3n^2 + 4n - \eta.n - \eta - 2)$$
$$= 2n + n^2 \times (n-1) + (n^3 - 3n^2 + 4n - \eta.n + \eta - 2)$$
$$= 2n^3 - n^2(4+\alpha) + n(6+\alpha) - 2,\ 0 < \alpha = \frac{\eta}{n} \leq 1 \quad (4)$$

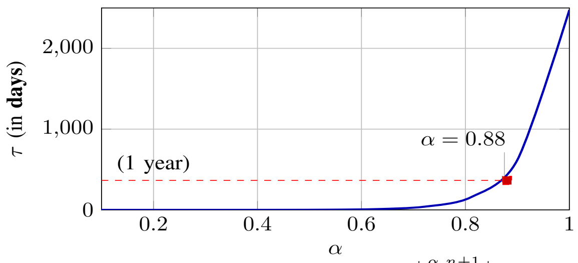

Fig. 4. Estimated decryption time ($\tau = \tau_0 * 2^{\lfloor \frac{\alpha.n+1}{3} \rfloor}$) for b04 (the smallest circuit under consideration for scrambling), as a function of area-cost constraint $\alpha = \frac{\eta}{n}$ $(0 < \alpha \leq 1)$. Please note the Y-axis range.

This suggests the worst-case *HP constraint* complexity is $\mathcal{O}(n^3)$ (because $\alpha \leq 1$). We have seen earlier that the connection constraint complexity is $\mathcal{O}(n)$ and combinational circuit constraint complexity is $\mathcal{O}(g) = \mathcal{O}(n)$, because the ratio of flip-flops to gates lies in a restricted range. Thus, the worst-case total CNF reduction complexity is $\mathcal{O}(n^3)+\mathcal{O}(n)+\mathcal{O}(n) = \mathcal{O}(n^3)$.

The last-but-one column in Table II shows the practical impossibility to launch the SAT-based attack on the scrambled instances, hence we report the decryption time per iteration in the fourth column of this table. For b19 processor, when scrambled with $\eta = n$ results in only 0.1% overhead, but we notice 91 $hrs.$ decryption time per iteration and a total of $10^{100.2}$ iterations needed to decrypt the scrambling key. This causes the estimated decryption time to be $10^{98.2}$ years, thus demonstrating the power of the proposed technique. Further, Figure 4 shows exponential increase in the estimated decryption time as a function of α. The last column in Table II shows the overheads of scrambling. The overhead decreases with an increase in circuit complexity, demonstrating the scalability of the proposed technique. Please note that here, $\alpha = 1$ is used i.e. all the SFFs were used for scrambling, yet the area overhead is acceptable for large designs. Thus, the overheads will be further less for smaller values of α.

IV. DISCUSSION

This section identifies several aspects that are orthogonal to our proposed research.

A. Modified SAT attack using a random/wrong scan key

Although the objective of the attacker is to find the correct functional key and not the scan key, it is important to obtain the correct scan key to be able to proceed with the attack.

One might consider generating a random/wrong scan key and proceeding with the SAT attack [3].

It is possible to generate a random/wrong scan key and use it in the CNF, but it is not possible to use this wrong scan key in the activated chip. This is because *activated* by definition implies the correct key (functional + scan) is already applied to the chip through a tamper-proof memory. Since it is not possible to use the wrong key in the activated chip, using this modified version of SAT attack will be unable to find the correct functional key.

B. Related work on combinational locking techniques

Stripped-Functionality Logic-Locking (SFLL) [15] was only scheme that was broadly resilient to attacks, yet it recently failed against functional analysis of logic-locking (FALL) [16] and SMT [17] attacks. SFLL provides provable security guarantees and is the first version of a class of techniques known as provably-secure logic locking (PSLL). Overall, it has been a cat and mouse game with combinational defenses and attacks, more recently this trend was also observed in PSLL.

$SFLL{-}HD^h$ [15] has the property that all the protected cubes are of identical Hamming distance to the secret key. Attacks including FALL [16], SFLL-hd-Unlock [18], and GNNUnlock [19] exploit this feature and/or the structural traces left by the locking algorithm in the functionality-stripped circuit (FSC) portion of the design. On the other hand, in $SeqL+$ distinguishing the correct scan-chain using structural analysis is computationally intractable for the adversary, hence it is not possible to perform functional analysis by applying inputs and observing outputs. because it is not possible to apply inputs and observe outputs through scan-chains, which are scrambled.

Attacks like FALL [16], SFLL-hd-Unlock [18], and GNNUnlock [19] could not break $SFLL{-}flex^{cxk}$, due to the ability of the user to specify arbitrary input cubes, which are independent, and harder to identify. However, $SFLL{-}flex^{cxk}$ also leaves structural traces in the FSC, similar to $SFLL{-}HD^h$, and has recently been broken by `Valkyrie` [20] using advanced critical signal identification through structural analysis, generating fault-pruning input patterns (FPIPs) using automatic test pattern generation (ATPG), and simulation based comparison with oracle responses to prune incorrect solutions.

The vulnerability of pre-SFLL approaches has already been discussed in the SFLL paper. All of the post-SFLL defenses which are part of the cat-and-mouse game in PSLL were broken by `Valkyrie` [20]. $SeqL+$ is however resilient to `Valkyrie` because it deploys scan cell scrambling and does not modify the combinational logic to improve corruptibility. As a result, critical signal analysis is not useful to identify the correct sequence of flip-flops in the scrambled scan-chain.

Furthermore, the average area overhead for the largest five benchmarks in $SFLL{-}flex^{cxk}$ is $\approx 6\%$ [15], while it is only $\approx 0.2\%$ in case of $SeqL+$ on average for the largest five benchmarks. Finally, it should be noted that ours is not the first paper on scan-obfuscation, we have identified the shortcomings in prior approaches and addressed them as shown earlier in Table I.

V. Conclusions and Future Work

We have proposed *SeqL+*, a defense, that embeds exponentially many number of *Hamiltonian Paths* into the *scramble digraph* thereby thwarting *SAT-based attacks*. We have shown both the theoretical and empirical improvements in the security of *SeqL+* compared to the state-of-the-art scan-obfuscation schemes. Since we scramble only the security-chain, it is area-efficient. The scalability demonstrates applicability to mainstream industry practice. We have also demonstrated that the method is computationally-secure and it is possible to trade-off overheads with security. For small circuits, our overheads are relatively more costly compared to EFF [2]. Since we do not have proof of optimality for our proposed defense, the overheads can be reduced further with future extensions.

VI. Acknowledgment

This project is funded in part by the NC State Faculty Research and Professional Development Program.

References

[1] D. Zhang et al, “Dynamically obfuscated scan for protecting IPs against scan-based attacks throughout supply chain,” in *IEEE VTS*, 2017, pp. 1–6.
[2] R. Karmakar et al, “A Scan Obfuscation Guided Design-for-Security Approach For Sequential Circuits,” *IEEE TCAS II*, pp. 1–1, 2019.
[3] L. Alrahis et al, “ScanSAT: Unlocking Static and Dynamic Scan Obfuscation,” *IEEE TETC*, pp. 1–1, 2019.
[4] R. Karmakar et al, “Improving Security of Logic Encryption in Presence of Design-for-Testability Infrastructure,” in *IEEE ISCAS*, 2019, pp. 1–5.
[5] A. Cui et al, “A Guaranteed Secure Scan Design Based on Test Data Obfuscation by Cryptographic Hash,” *IEEE TCAD*, vol. 39, no. 12, pp. 4524–4536, 2020.
[6] R. Karmakar et al, “Efficient Key-Gate Placement and Dynamic Scan Obfuscation Towards Robust Logic Encryption,” *IEEE TETC*, vol. 9, no. 4, pp. 2109–2124, 2021.
[7] M. S. Rahman et al, “Security Assessment of Dynamically Obfuscated Scan Chain Against Oracle-Guided Attacks,” *ACM TODAES*, vol. 26, no. 4, 2021.
[8] X. Wang et al, “Secure Scan and Test Using Obfuscation Throughout Supply Chain,” *IEEE TCAD*, vol. 37, no. 9, pp. 1867–1880, 2018.
[9] U. Guin et al, “Robust design-for-security architecture for enabling trust in IC manufacturing and test,” *IEEE TVLSI*, vol. 26, no. 5, pp. 818–830, 2018.
[10] H. M. Kamali et al, “On designing secure and robust scan chain for protecting obfuscated logic,” in *IEEE GLSVLSI*, 2020, p. 217–222.
[11] S. Potluri et al, “SeqL: Secure Scan-Locking for IP Protection,” in *ISQED*, 2020, pp. 7–13.
[12] N. Limaye et al, “Is robust design-for-security robust enough? attack on locked circuits with restricted scan chain access,” in *IEEE ICCAD*, 2019.
[13] K. Z. Azar et al, “NNgSAT: Neural Network guided SAT Attack on Logic Locked Complex Structures,” in *IEEE ICCAD*, 2020, pp. 1–9.
[14] P. Subramanyan, S. Ray, and S. Malik, “Evaluating the security of logic encryption algorithms,” in *IEEE HOST*, 2015, pp. 137–143.
[15] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, “Provably-secure logic locking: From theory to practice,” in *ACM CCS*, 2017, pp. 1601–1618.
[16] D. Sirone et al, “Functional analysis attacks on logic locking,” in *IEEE/ACM DATE*, 2019, pp. 936–939.
[17] K. Z. Azar et al, “SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks,” in *CHES*, 2019.
[18] F. Yang, M. Tang, and O. Sinanoglu, “Stripped Functionality Logic Locking With Hamming Distance-Based Restore Unit (SFLL-hd) – Unlocked,” *IEEE TIFS*, vol. 14, no. 10, pp. 2778–2786, 2019.
[19] L. Alrahis et al, “GNNUnlock: Graph Neural Networks-based Oracle-less Unlocking Scheme for Provably Secure Logic Locking,” in *IEEE/ACM DATE*, 2021, pp. 780–785.
[20] N. Limaye et al, “Valkyrie: Vulnerability Assessment Tool and Attack for Provably-Secure Logic Locking Techniques,” *IEEE TIFS*, vol. 17, pp. 744–759, 2022.